



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
[www.uspto.gov](http://www.uspto.gov)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/772,207	02/03/2004	Sebastian Lange	MS1-1823US	4217
22801	7590	05/22/2008	EXAMINER	
LEE & HAYES PLLC			DOAN, TRANG T	
421 W RIVERSIDE AVENUE SUITE 500			ART UNIT	PAPER NUMBER
SPOKANE, WA 99201			2131	
			MAIL DATE	DELIVERY MODE
			05/22/2008	PAPER

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

<b>Office Action Summary</b>	<b>Application No.</b>	<b>Applicant(s)</b>	
	10/772,207	LANGE ET AL.	
	<b>Examiner</b>	<b>Art Unit</b>	
	TRANG DOAN	2131	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

#### Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

#### Status

- 1) Responsive to communication(s) filed on 07 February 2008.
- 2a) This action is **FINAL**.                    2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

#### Disposition of Claims

- 4) Claim(s) 1-9, 11-15, 17-22, 24-34, 36-43 and 45-50 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) Claim(s) \_\_\_\_\_ is/are allowed.
- 6) Claim(s) 1-9, 11-15, 17-22, 24-34, 36-43 and 45-50 is/are rejected.
- 7) Claim(s) \_\_\_\_\_ is/are objected to.
- 8) Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

#### Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on 23 February 2004 is/are: a) accepted or b) objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

#### Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) All    b) Some \* c) None of:
1. Certified copies of the priority documents have been received.
  2. Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

#### Attachment(s)

- |  |   |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)          | 4) <input type="checkbox"/> Interview Summary (PTO-413)           |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | Paper No(s)/Mail Date. _____ .                                    |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)          | 5) <input type="checkbox"/> Notice of Informal Patent Application |
| Paper No(s)/Mail Date _____ .  | 6) <input type="checkbox"/> Other: _____ .                        |

## **DETAILED ACTION**

1. This action is in response to the amendment filed on 02/07/2008.
2. Claims 1, 11-13, 18-22, 24-27, 30, 36, 38, 40 and 41 have been amended.
3. Claims 10, 16, 23, 35 and 44 have been canceled.
4. Claims 46-50 have been added.
5. Claims 1-9, 11-15, 17-22, 24-34, 36-43 and 45-50 are pending for consideration.

### ***Response to Arguments***

6. Examiner withdraws the 101 rejections in the previous Office action because the claims have been amended.
7. Examiner also withdraws the 112, 2<sup>nd</sup> paragraph, rejections because Applicant's arguments are persuasive.
8. Applicant's arguments filed on 02/07/2008 have been fully considered but they are not persuasive.

Applicant argues on pages 23-25 of the Remarks that Koved fails to disclose finding a set of required permissions for each execution path by one or more simulated stack walks that each include a plurality of the assemblies, wherein each call in each execution path has a correspond permissions set. Examiner respectfully disagrees. Koved does teach finding a set of required permissions for each execution path by one or more simulated stack walks that each include a plurality of the assemblies, wherein each call in each execution path has a correspond permission set (See Koved on page 2, column 2 and last paragraph: "a formalization of stack introspection, which examines

authorization based on the principals (signers and/or origin of the code) currently active in a Thread stack at run-time, as is found in Java ... at each stack frame, as well as the result of any authorization test encountered" and page 3, column 1 and first paragraph: "the runtime stack ... to discover authorization requirements by analyzing all possible paths through the program").

Regarding claims, 2-9, 11, 13-15, 17, 18, 20-22, 24-25, 27-34, 36, 37, 39, 40, 41, 42-43 and 45, Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

The examiner is not trying to teach the invention but is merely trying to interpret the claim language in its broadest and reasonable meaning. The examiner will not interpret to read narrowly the claim language to read exactly from the specification, but will interpret the claim language in the broadest reasonable interpretation in view of the specification. Therefore, the examiner asserts that cited prior art does teach or suggest the subject matter broadly recited in independent Claims and in subsequent dependent Claims. Accordingly, rejections for claims 1-9, 11-15, 17-22, 24-34, 36-43 and 45-50 are respectfully maintained.

***Claim Rejections - 35 USC § 102***

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

10. Claims 1-9, 11-15, 17-22, 24-34, 36-43 and 45-50 are rejected under 35 U.S.C. 102(b) as being anticipated by Koved et al. (reference U) (hereinafter Koved).

Regarding claim 1, Koved discloses simulating the execution of all execution paths of one or more assemblies in managed code, wherein the managed code is a managed shared library or an executable (Koved: on page 1, column 2, under INTRODUCTION heading, second paragraph: “developer reads ... libraries used (including the Java run-time libraries”) and reduces the required access rights); and finding a set of required permissions for each execution path by one or more simulated stack walks that each include a plurality of the assemblies, wherein each call in each execution path has a corresponding said permissions set (Koved: on page 2, column 2 and last paragraph: “a formalization of stack introspection, which examines authorization based on the principals (signers and/or origin of the code) currently active in a Thread stack at run-time, as is found in Java ... at each stack frame, as well as the result of any authorization test encountered” and page 3, column 1 and first paragraph: “the runtime stack ... to discover authorization requirements by analyzing all possible paths through the program”).

Regarding claim 2, Koved discloses wherein the execution paths for only one said assembly in managed code are simulated to find the set of required permissions for each said execution path by a union of the permissions for each said execution path (Koved: on page 2, column 2, third paragraph; page 3, column 1, first paragraph; and

page 3 and page 4, under Authorization Model section: “In this paper...an invocation graph and data flow analysis...more accurate authorization information.” “Our approach...discover authorization requirements by analyzing all possible paths through the program.” “It can be seen...the value of Required Permissions (n) (i.e., RP(n)) at the input to a node n...by means of a set of union operation”).

Regarding claim 3, Koved discloses wherein: the one or more assemblies in managed code correspond to an application (Koved: page 3, column 1, third paragraph: “Each Java application class...associated with a set of right, or privileges, granted to the code); and the set of required permissions for each said execution path comprises a union of the permissions for each said execution path (Koved: page 3, column 2; and page 4 column 2: “Since, in general...along paths towards nodes in Nstart. This process associates a set of required requirements RP(n) with each node n in Nstart. More precisely, it computes RP(n) for all n belong to N”. “It can be seen that the data flow...by means of a set union operation”).

Regarding claim 4, Koved discloses wherein: the assemblies in managed code correspond to a shared library (Koved: page 8, column 1, third paragraph: “For a given application or classes in a library...identify the set of Java 2 Permissions required for each class in the analysis scope”); and the set of required permissions for each said execution path comprises one separate permission set per entry point in the shared library (Koved: on page 1, under ABSTRACT section; and page 2, column 1, under Prior Work section: “This paper presents...compute at each program point the set of

access rights required by the code”...“authorization tests...to the current approach to defining authorization points within code”).

Regarding claim 5, this claim has limitations that is similar to those of claims 2 and 3, thus it is rejected with the same rationale applied against claims 2 and 3 above.

Regarding claim 6, Koved discloses wherein one of more of the calls in at least one said execution path is an cross assembly call (Koved: on page 2, column 2, third paragraph: “In the aforementioned works...Java runtime calls one of the SecurityManager authorization methods...to correctly identify authorization requirements”).

Regarding claim 7, Koved discloses wherein: the managed code is built to make use of a common language runtime (on page 2, column 2, third paragraph: “In the aforementioned works...Java runtime calls one of the SecurityManager authorization methods...to correctly identify authorization requirements”); each said assembly is packaged as an executable entity or as a data link library entity and each said assembly includes one or more methods (Koved: on page 1, under ABSTRACT section; and page 7, column 2, second and third paragraph: “The tool...to identify the access rights requirements for the product to enable it to run using Java 2 security model”).

Regarding claim 8, Koved discloses wherein the simulation of the execution of each said execution path comprises a simulation of the flow of argument data using intra and extra method data flow analysis for each said method (Koved: on page 2, column 1, second and third paragraph; and page 6, column 2, second paragraph: “To summarize...We present a context sensitive, flow sensitive analysis for computing the

access rights requirements of a program.” “To minimize conservativeness...the order of execution of instructions both intra- and inter procedurally thus improving the accuracy of the resulting graph”).

Regarding claim 9, Koved discloses wherein when the executable has permissions to execute that are not less than a union of permission sets for each said execution path, any dynamic execution of the executable will not trigger a security exception (Koved: on page 3, under Authorization Model-Access Rights Invocation Graph section; page 7, column 1, second paragraph; and page 8, column 1, first paragraph: “Performance is improved...NullPointerException in package...”).

Regarding claim 11, Koved discloses a computer readable storage medium having a tangible component including machine readable instructions for implementing the method as defined in Claim 1 (Koved: on page 4, column 1, third paragraph).

Regarding claim 12, this claim has limitations that is similar to those of claims 1-6, thus it is rejected with the same rationale applied against claims 1-6 above.

Regarding claim 13, Koved discloses wherein the managed code environment comprises: a managed code portion including: the assemblies (Koved: see ABSTRACT section); and a virtual machine (Koved: on page 3, column 1, paragraph 3: “Each Java application...the Java Virtual Machine...to the code”); a native code portion including: an execution engine for the virtual machine (Koved: see ABSTRACT section: “Java...protects systems...execute the code...in deployed systems”); and an operating system under the execution engine (Koved: see ABSTRACT section: “Java...protects systems...execute the code...in deployed systems”).

Regarding claim 14, this claim has limitations that is similar to those of claim 7, thus it is rejected with the same rationale applied against claim 7 above.

Regarding claim 15, this claim has limitations that is similar to those of claim 9, thus it is rejected with the same rationale applied against claim 9 above.

Regarding claim 17, Koved discloses wherein the managed code environment enforces partial trust security contexts (Koved: on page 3, column 1, 2 paragraph: “Rather than analyzing...enforce specific security policies...updated to enable the code to execute”).

Regarding claim 18, this claim has limitations that is similar to those of claim 11, thus it is rejected with the same rationale applied against claim 11 above.

Regarding claim 19, Koved discloses instructions that, when executed, perform a simulation of the execution of every data and control flow for managed code from which an estimate is derived of the minimum security requirements needed to dynamically execute the managed code without triggering a security exception, wherein the simulation of the execution comprises, for each data and control flow for the managed code, one or more simulated stack walks that include two or more the assemblies (Koved: on page 1, column 1; page 7 under Generation of a Security Policy Decryption section; page 1, column 2, 2<sup>nd</sup> and 3<sup>rd</sup> paragraph: “ This paper presents...computing the access rights requirements”; page 2, column 2 and last paragraph: “a formalization of stack introspection, which examines authorization based on the principals (signers and/or origin of the code) currently active in a Thread stack at run-time, as is found in Java ... at each stack frame, as well as the result of any authorization test encountered”

and page 3, column 1 and first paragraph: “the runtime stack ... to discover authorization requirements by analyzing all possible paths through the program”).

Regarding claim 20, this claim has limitations that is similar to those of claim 7 and 14, thus it is rejected with the same rationale applied against claims 7 and 14 above.

Regarding claim 21, this claim has limitations that is similar to those of claim 13, thus it is rejected with the same rationale applied against claim 13 above.

Regarding claim 22, this claim has limitations that is similar to those of claim 20, thus it is rejected with the same rationale applied against claim 20 above.

Regarding claim 24, this claim has limitations that is similar to those of claim 16, thus it is rejected with the same rationale applied against claim 16 above.

Regarding claim 25, this claim has limitations that is similar to those of claim 17, thus it is rejected with the same rationale applied against claim 17 above.

Regarding claim 26, Koved discloses means for processing; means for storing information in memory coupled to the means for processing; virtual machine means, stored in the memory, in a managed code portion, for operating a plurality of assemblies in managed code, wherein the managed code is a managed shared library or an executable and is in the managed code portion; execution engine means, in a native code portion, for executing the virtual machine means (Koved: on page 3, column 1, third paragraph: “Each Java application class...Java Virtual Machine...privileges, granted to the code”); means, in the native code portion, for providing an operating system (Koved: on page 3, column 1, third paragraph; and page 4, column 1, first

paragraph: "Each Java application class...Java Virtual Machine...privileges, granted to the code"); means for making a call in the managed code portion for access by one assembly to another assembly for which a permissions set is required (Koved: see ABSTRACT section on page 1); means in the managed code portion for gathering the permissions set from each said call (Koved: on page 4, column 1, second paragraph); means for deriving a union of the gathered permissions sets (Koved: on page 3 under Authorization Model-Access Rights Invocation Graph section); and means in the managed code portion for simulating the execution of all possible execution paths for the managed shared library or the executable to derive therefrom the derived union of the gathered permissions sets wherein the means for simulating the execution performs, for each execution path, one or more simulated stack walks that each include a plurality of assemblies (Koved: on page 3 under Authorization Model-Access Rights Invocation Graph section; page 2, column 2 and last paragraph: "a formalization of stack introspection, which examines authorization based on the principals (signers and/or origin of the code) currently active in a Thread stack at run-time, as is found in Java ... at each stack frame, as well as the result of any authorization test encountered" and page 3, column 1 and first paragraph: "the runtime stack ... to discover authorization requirements by analyzing all possible paths through the program").

Regarding claim 27, Koved discloses means for compiling the assemblies from an intermediate language code and metadata into native code; and means for loading the native code with a Common Language Runtime loader in the native code portion to load the compiled native code, wherein the execution engine means executes the

compiled native code in the native code portion (Koved: on page 3, column 1, first paragraph: “Permission.implies...test cases”).

Regarding claim 28, Koved discloses wherein the managed code portion further comprises one or more files associated with user code that, when compiled into an intermediate language code and metadata generated by a language compiler, are represented by the assemblies (Koved: on page 3, column 1, third paragraph: “Each Java application class...Java Virtual Machine...privileges, granted to the code”).

Regarding claim 29, Koved discloses wherein the execution engine means in the native code portion further comprises a compiler to compile each said assembly into native code for execution by the native code portion (Koved: on page 3, column 1, third paragraph: “Each Java application class...Java Virtual Machine...privileges, granted to the code”).

Regarding claim 30, Koved discloses wherein the execution engine means in the native code portion further comprises: a Just In Time compiler to compile each said assembly into native code; and a common language runtime loader to load the compiled native code for execution by the native code portion (on page 3, column 1, third paragraph: “Each Java application class...Java Virtual Machine...privileges, granted to the code”).

Regarding claim 31, Koved discloses means, in the native code portion, for forming a response to the call; and means for returning the response to the first assembly in the managed code portion (Koved: on page 3, column 1, third paragraph;

and page 4, column 1, first paragraph: "Each Java application class...Java Virtual Machine...privileges, granted to the code").

Regarding claim 32, Koved discloses wherein: the managed code is built to make use of a common language runtime; each said assembly is packaged as an executable entity or as a data link library entity; and each said assembly includes one or more methods (Koved: on page 1, under ABSTRACT section; and page 7, column 2, second and third paragraph: "The tool...to identify the access rights requirements for the product to enable it to run using Java 2 security model").

Regarding claim 33, Koved discloses wherein the simulation of the execution comprises, for each said execution path, a simulation of the flow of argument data using intra and extra data flow analysis for each said method (Koved: on page 2, column 1, second and third paragraph; and page 6, column 2, second paragraph: "To summarize...We present a context sensitive, flow sensitive analysis for computing the access rights requirements of a program." "To minimize conservativeness...the order of execution of instructions both intra- and inter procedurally thus improving the accuracy of the resulting graph").

Regarding claim 34, this claim has limitations that is similar to those of claim 9, thus it is rejected with the same rationale applied against claim 9 above.

Regarding claim 36, Koved discloses wherein each call in each simulated stack walk has a corresponding permissions set (Koved: on page 3 under Authorization Model-Access Rights Invocation Graph section: "For any node...set of required Permissions for n").

Regarding claim 37, this claim has limitations that is similar to those of claim 17, thus it is rejected with the same rationale applied against claim 17 above.

Regarding claim 38, this claim has limitations that is similar to those of claim 26, thus it is rejected with the same rationale applied against claim 26 above.

Regarding claim 39, this claim has limitations that is similar to those of claims 27 and 28, thus it is rejected with the same rationale applied against claims 27 and 28 above.

Regarding claim 40, this claim has limitations that is similar to those of claim 30, thus it is rejected with the same rationale applied against claim 30 above.

Regarding claim 41, this claim has limitations that is similar to those of claim 22, thus it is rejected with the same rationale applied against claim 22 above.

Regarding claim 42, this claim has limitations that is similar to those of claim 8, thus it is rejected with the same rationale applied against claim 8 above.

Regarding claim 43, this claim has limitations that is similar to those of claims 9 and 15, thus it is rejected with the same rationale applied against claims 9 and 15 above.

Regarding claim 45, this claim has limitations that is similar to those of claim 17, thus it is rejected with the same rationale applied against claim 17 above.

Regarding claim 46, Koved discloses wherein the simulated stack walk comprises: entering a public entry point of a method in the assembly; gathering a permission set for the method; determining whether the method calls another method; gathering a permission set for the called method; and creating a union of the gathered

permission sets (Koved: on page 2, column 2 and last paragraph: “a formalization of stack introspection, which examines authorization based on the principals (signers and/or origin of the code) currently active in a Thread stack at run-time, as is found in Java … at each stack frame, as well as the result of any authorization test encountered”; page 3, column 1 and first paragraph: “the runtime stack … to discover authorization requirements by analyzing all possible paths through the program”; and page 3 section 3 “Authorization Model – Access Rights Invocation Graph”).

Regarding claim 47, Koved discloses wherein the simulated stack walk comprises: entering a public entry point of a method in the assembly; gathering a permission set for the method; determining whether the method calls another method; for each called method: gathering a permission set for the called method; and determining whether the called method calls a subsequent method; and creating a union of the gathered permission sets (Koved: on page 2, column 2 and last paragraph: “a formalization of stack introspection, which examines authorization based on the principals (signers and/or origin of the code) currently active in a Thread stack at run-time, as is found in Java … at each stack frame, as well as the result of any authorization test encountered”; page 3, column 1 and first paragraph: “the runtime stack … to discover authorization requirements by analyzing all possible paths through the program”; and page 3 section 3 “Authorization Model – Access Rights Invocation Graph”).

Regarding claim 48, this claim has limitations that is similar to those of claim 47, thus it is rejected with the same rationale applied against claim 47 above.

Regarding claim 49, Koved discloses wherein the one or more simulated stack walks comprise: means for entering a public entry point of a method in the assembly; means for gathering a permission set for the method; means for determining whether the method calls another method; for each called method: means for gathering a permission set for the called method; means for determining whether the called method calls a subsequent method; and means for repeating the previous gathering and determining until any gathered permission set is duplicative; and means for creating a union of the gathered permission sets (Koved: on page 2, column 2 and last paragraph: “a formalization of stack introspection, which examines authorization based on the principals (signers and/or origin of the code) currently active in a Thread stack at runtime, as is found in Java ... at each stack frame, as well as the result of any authorization test encountered”; page 3, column 1 and first paragraph: “the runtime stack ... to discover authorization requirements by analyzing all possible paths through the program”; and page 3 section 3 “Authorization Model – Access Rights Invocation Graph”).

Regarding claim 50, this claim has limitations that is similar to those of claim 49, thus it is rejected with the same rationale applied against claim 49 above.

### ***Conclusion***

11. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Kershenbaum et al. discloses method and apparatus for automatically determining optimum placement of privileged code locations in existing code (US 2004/0040017).
- Stefk et al. discloses method and apparatus for executing code in accordance with usage rights (US 2003/0225698).
- Anand et al. discloses flexible and dynamic derivation of permissions (US 6044466).
- Apperson et al. discloses system and method for safety distributing executable objects (US 5978484).
- Lamacchia et al. discloses applying a permission grant set to a call stack during runtime (US 7076557).
- Scheifler et al. discloses stack-based access control using code and execution identifiers (US 6138238).

12. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the

Art Unit: 2131

shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to TRANG DOAN whose telephone number is (571)272-0740. The examiner can normally be reached on Monday-Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ayaz Sheikh can be reached on (571) 272-3795. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Trang Doan/  
Examiner, Art Unit 2131  
/Ayaz R. Sheikh/  
Supervisory Patent Examiner, Art Unit 2131